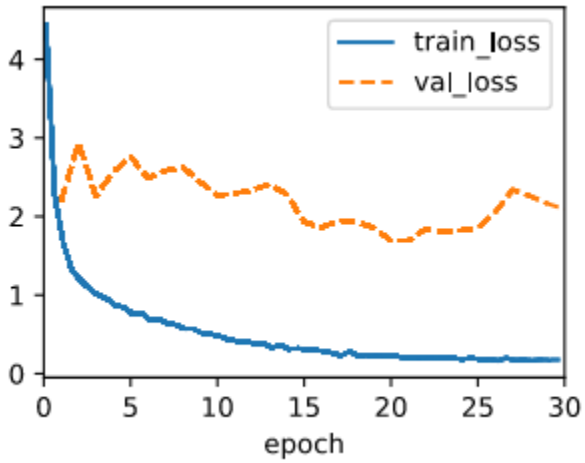
ECGR-5103: Homework 6
Patrick Flynn | ID: 801055057
28 April 2023
Github: https://github.com/pflynn157/ecgr-5106

Problem 1

In this problem, we trained a transformer to perform natural language translation tests. The code of this consisted of several parts. In the first part, we set up the feed-forward neural network and the normalization layers. Next, we created an encoder block, and from that the encoder for the actual transformer. The decoder was also similar. We created the decoder block, followed by the actual decoder, the latter for the actual transformer. Finally, we connected it together and formed the final transformer.

In the experiments that followed, we made several adjustments to the parameters to make the transformer deeper, thereby seeing what effect each setting would have on performance.

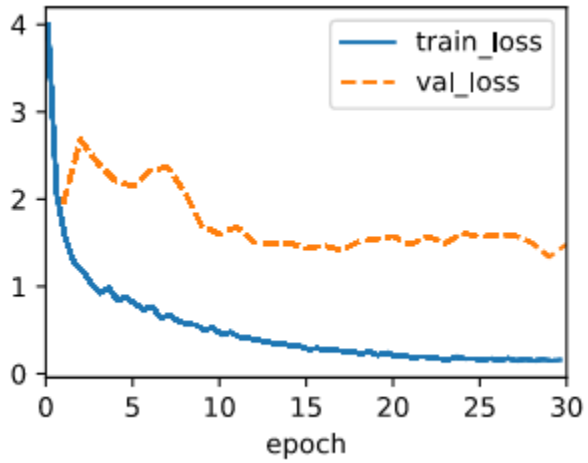
The baseline model had 256 hidden layers, 2 blocks, a dropout rate of 0.2, 64 feed-forward network hidden layers, and 4 attention heads. The training results were reasonable. The training loss was low, though the validation loss stayed generally constant.



The test translation results were fairly good. Of the four tests, the bleu results were 1.0 for three of them, and still high for the third.

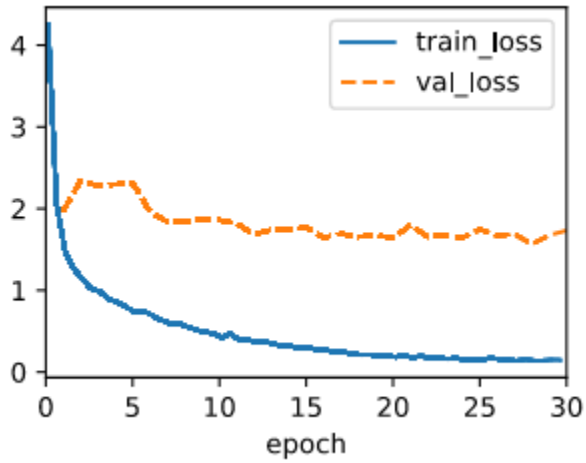
```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mort', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

In the first experiment, we adjusted the network by increasing the number of feed-forward network layers. The training loss stayed roughly the same, but with an improvement in the validation loss. The test results were still good for three of them, but one of the translations did even worse than before.



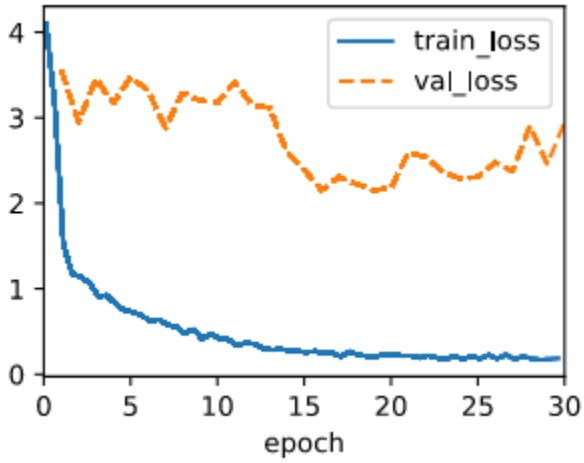

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Next, we adjusted the number of feed-forward network layers even more. This didn't create any huge changes- the training loss was still about the same, but the validation loss was more constant, and even seemed to slowly improve. The test results were as good as the baseline test:



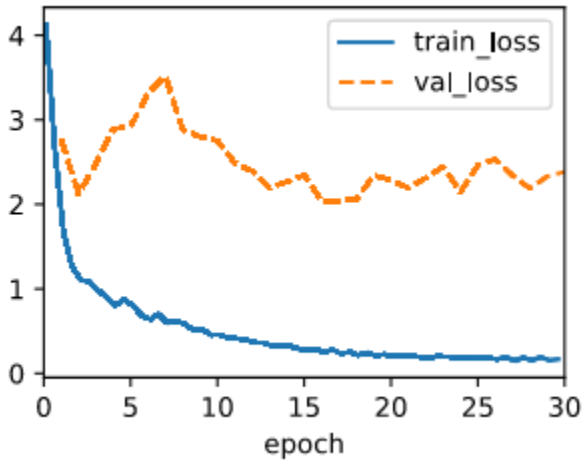

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Next, we adjusted the number of hidden layers. Again, the training loss was still roughly the same, but the validation loss was much worse. The test results, however, were still good- the same as the baseline.



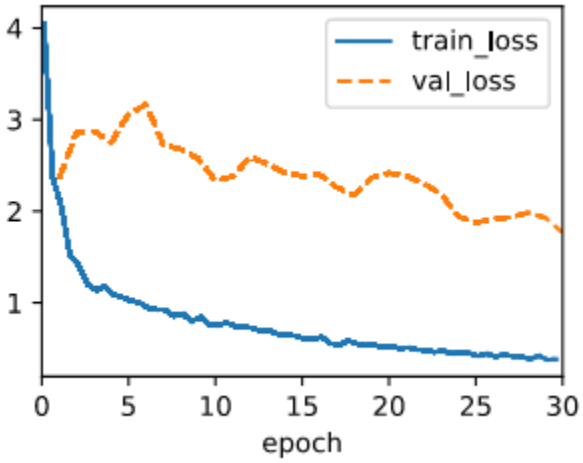

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Next, we adjust the number of blocks. The results here were similar to that of adjusting the number of hidden layers. The training loss was still good, but the validation loss was still all over the place, and generally worse than the baseline- though certainly not as bad as our last test. The test results were the same.



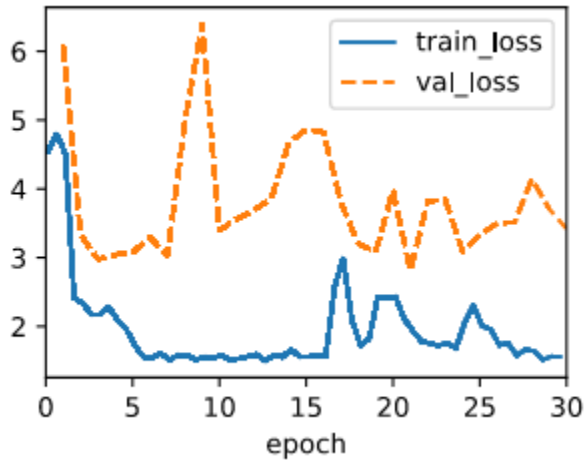

```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Next, we tried adjusting the dropout. This really doesn't help- pretty much everything was worse. The training and validation loss were both higher, and the test results were worse.




```
go . => ['va', 'doucement', '!'], bleu,0.000
i lost . => ['je', 'suis', '<unk>', 'perdu', '.'], bleu,0.447
he's calm . => ['il', 'est', 'calme', '.'], bleu,1.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Finally, we tried combining it altogether- adjusting the parameters with every adjustment we made in the previous experiments. The results were absolutely terrible. The training loss and validation loss were high, with the validation loss in particular being all over the place. The test results were all 0's.



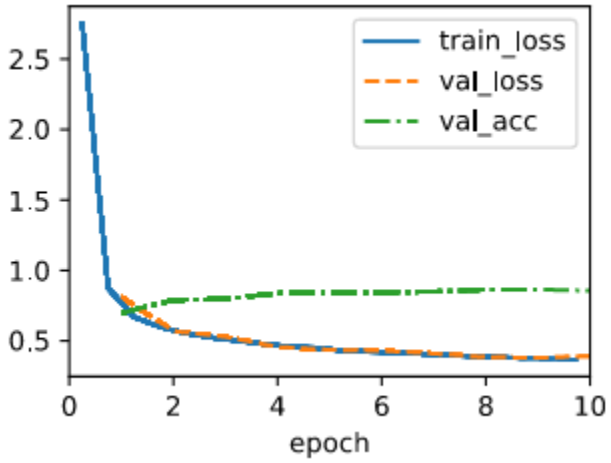

```
go . => [], bleu,0.000
i lost . => [], bleu,0.000
he's calm . => [], bleu,0.000
i'm home . => [], bleu,0.000
```

Conclusion: The baseline was good, but adjusting the feed-forward neural network seemed to yield the best results. Since adjusting this basically affects how much data can be processed at once, this makes sense that it would yield better results. In no case did the translations significantly improve, but the validation loss did improve the most with these adjustments.

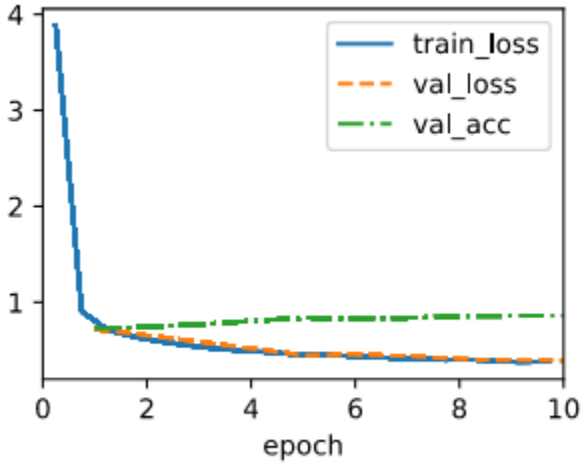
Problem 2

In this problem we built, trained, and experimented with a vision transformer. The overall process was similar to that of our language transformer in the previous problem. For this transformer, we used the FashionMNIST dataset.

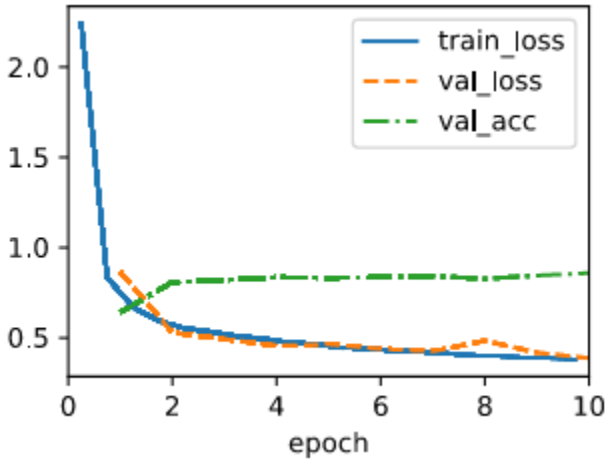
The baseline model in this experiment seemed to perform rather well. Training and validation loss was consistent, and while the validation accuracy wasn't huge, it was reasonably good.



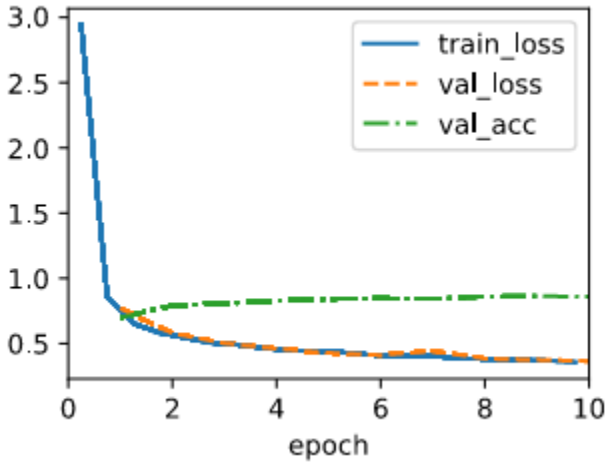
In the first experiment, we adjusted the number of hidden layers. While it didn't affect the training or validation loss much, it did make the validation accuracy worse.



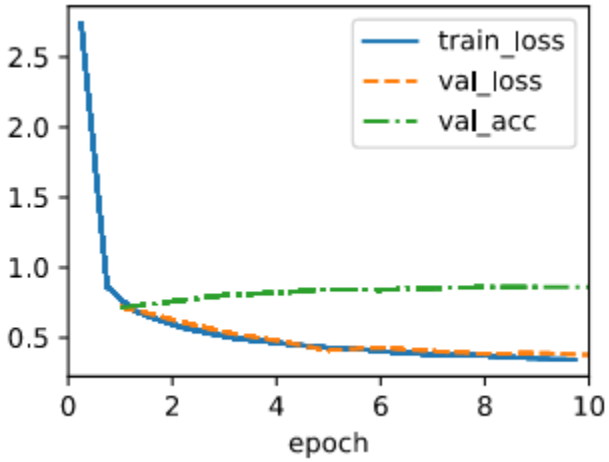
Next, we adjusted the number of MLP hidden layers. This actually yielded rather good results. The training and validation loss were slightly worse, but the validation accuracy was noticeably better, even over the baseline.



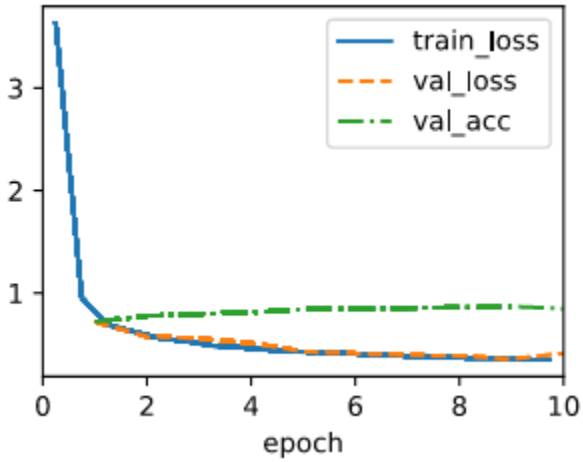
Next, we adjusted the number of attention heads. This really didn't yield any changes over the baseline model.



Adjusting the number of blocks had a similar result, or perhaps the lack thereof. Once again, the loss and accuracy along all metrics was similar to the baseline.



Finally, we combined all our changes together to see what effect that would have. In general, this really didn't make a huge difference. The results were good, but still roughly similar to the baseline.



Conclusion: Making adjustments to the vision transformer really doesn't seem to yield dramatically different results. Changing the MLP layers yielded the most noticeable results, but even here, it was still roughly the same.